Djillali Liabes University of Sidi Bel Abbes
Faculty of Exact Sciences
Department of Computer Sciences

Module : Modélisation et Simulation
1st year of Master's degeree in
Networks,Information Systems & Security(RSSI)
2021/2022

Chane de Markov
TP-02

Students:
Hadjazi M.Hisham
Amuer Wassim
Group: 01 / RSSI

Instructor:
Dr. S.Benbekriti

A paper submitted in fulfilment of the requirements for the
Modélisation et Simulation TP-02

November 20, 2021

Contents

List of Figures ii

1 Solutions of Fiche TP-02 1
1.1 Exercise 1 2
1.1.1 Construire sous R une matrice stochastique 3x3 2
Calculer A^5 , A^{20} et A^{40} . Que remarque-t-on ? 2
Trouver les valeurs propres de cette matrice 3
1.2 Exercise 2 4
1.2.1 Écrire une fonction qui permet de trouver les classes d'équivalences d'une chaîne de Markov à partir de sa matrice de transition P . 4
1.3 Exercise 3 5
1.3.1 Écrire une fonction qui à partir de la matrice de transition P , détermine la distri-bution invariante. 5

List of Figures

1.1 matrice stochastique 3x3 2
1.2 A^5 , A^{20} et A^{40} 3
1.3 Proper Values of Matrix A. 4
1.4 invariant Matrix 6

Chapter 1

Solutions of Fiche TP-02

Notes regarding this solution :

This solution and the executions of the code in it was done in the following machine :

- *PC* : Lenovo IdeaPad S210 8GB
- *OS* : Linux Mint 20.2 Cinnamon Kernel v.5.4.0-88
- *IDE* : RStudio 2021.09.0 Build 351
- *R Version* : 3.6.1 (2019-07-05)

During This TP we were solving the questions according to our understanding of them, we had some difficulties in understanding some of them for example while we ere solving question #3 , it asks to represent results as a histogram to the proportion of each color while a histogram is used to represent the distribution of the population or results which didn't make scene to us so we used a barchart as we saw it was more fit, and we used a histogram in question #6 to showcase where we think it is more appropriate to use it (of course to our limited understandings).

Also during the solutions we tried to get the data from experiments and not as facts from the question as it is much faster to calculate probability with formulas and simple calculations, but as the objective of this TP we thought that for our data we had to calculate the probability from experiments and simulations of the problems and we get results close to the calculations as the number of experiments increase.

We also included an R script file that has our code to ease the testing. The code we wrote has comments to explain what each piece of code does, we also included .tex in the document.

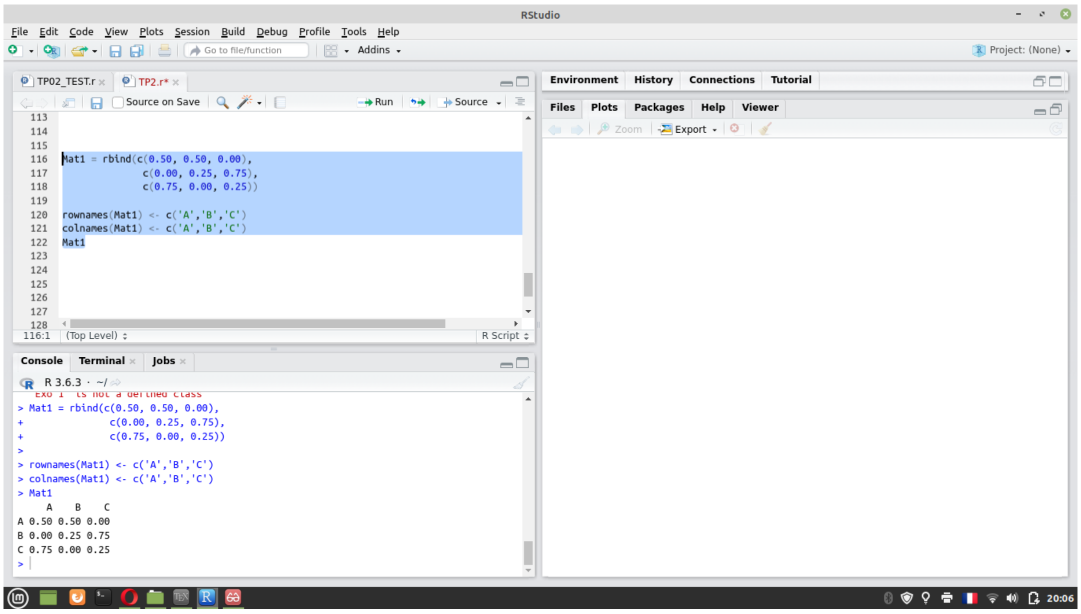
1.1 Exercise 1

1.1.1 Construire sous R une matrice stochastique 3x3

```r
Mat1 = rbind(c(0.50, 0.50, 0.00),
             c(0.00, 0.25, 0.75),
             c(0.75, 0.00, 0.25))

rownames(Mat1) <- c('A','B','C')
colnames(Mat1) <- c('A','B','C')
Mat1
```

The execution of the code gives the following matrix.



```
Mat1 = rbind(c(0.50, 0.50, 0.00),
             c(0.00, 0.25, 0.75),
             c(0.75, 0.00, 0.25))

rownames(Mat1) <- c('A','B','C')
colnames(Mat1) <- c('A','B','C')
Mat1
```

```
R 3.6.3 · ~/
  Exo 1  is not a defined class
> Mat1 = rbind(c(0.50, 0.50, 0.00),
+              c(0.00, 0.25, 0.75),
+              c(0.75, 0.00, 0.25))
>
> rownames(Mat1) <- c('A','B','C')
> colnames(Mat1) <- c('A','B','C')
> Mat1
     A    B    C
A 0.50 0.50 0.00
B 0.00 0.25 0.75
C 0.75 0.00 0.25
>
```

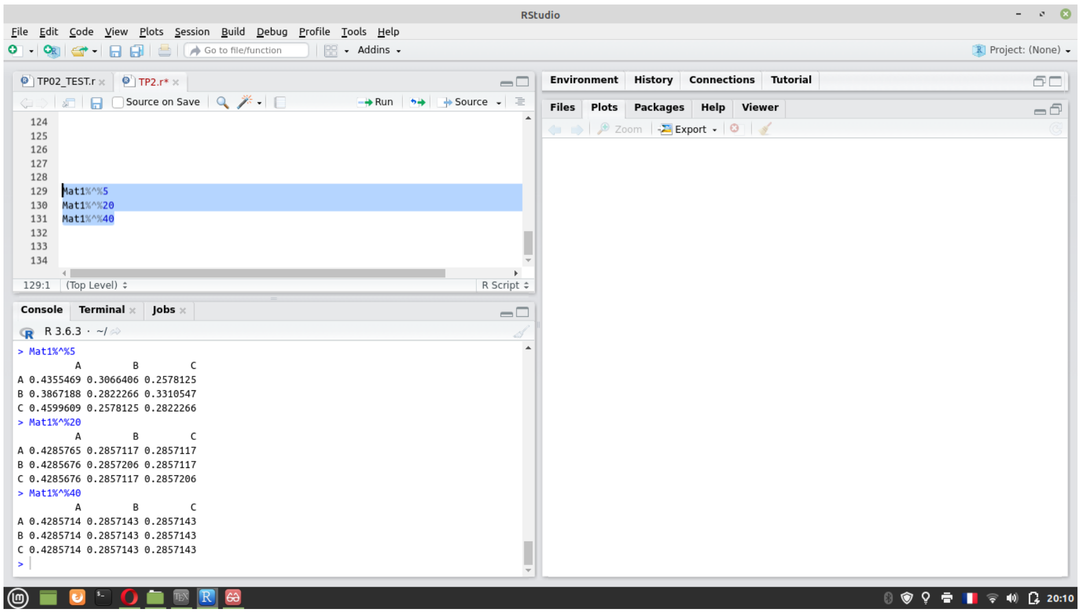
FIGURE 1.1: matrice stochastique 3x3

Calculer A^5 , A^{20} et A^{40} . Que remarque-t-on ?

by using the package **expm** we don't need to multiply the matrix in itself.

```r
Mat1%^%5
Mat1%^%20
Mat1%^%40
```

The execution of the code gives the following matrix.



```
Mat1%^%5
Mat1%^%20
Mat1%^%40
```

```
> Mat1%^%5
          A         B         C
A 0.4355469 0.3066406 0.2578125
B 0.3867188 0.2822266 0.3310547
C 0.4599609 0.2578125 0.2822266
> Mat1%^%20
          A         B         C
A 0.4285765 0.2857117 0.2857117
B 0.4285676 0.2857206 0.2857117
C 0.4285676 0.2857117 0.2857206
> Mat1%^%40
          A         B         C
A 0.4285714 0.2857143 0.2857143
B 0.4285714 0.2857143 0.2857143
C 0.4285714 0.2857143 0.2857143
>
```

FIGURE 1.2: A^5 , A^{20} et A^{40} .

After certain number of multiplying the matrix in itself, the matrix reaches a stable state where it stops changing and stays the same. at this stage we call it **Stable Matrix**.

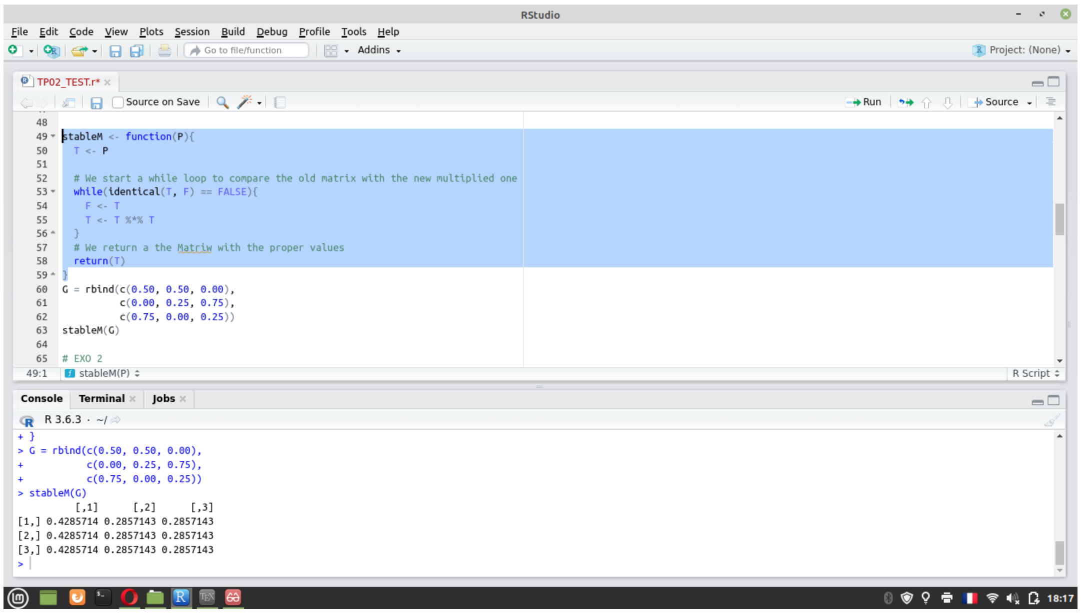
Trouver les valeurs propres de cette matrice

The stable matrix can be found after multiplying the matrix 30 times by itself.

```r
stableM <- function(P){
  T <- P

  # We start a while loop to compare the old matrix with the new
      multiplied one
  while(identical(T, F) == FALSE){
    F <- T
    T <- T %*% T
  }
  # We return a the Matriw with the proper values
  return(T)
}
```

The proper values of the matrix are as follow.



```
stableM <- function(P){
  T <- P

  # We start a while loop to compare the old matrix with the new multiplied one
  while(identical(T, F) == FALSE){
    F <- T
    T <- T %*% T
  }
  # We return a the Matriw with the proper values
  return(T)
}
G = rbind(c(0.50, 0.50, 0.00),
          c(0.00, 0.25, 0.75),
          c(0.75, 0.00, 0.25))
stableM(G)

# EXO 2
```

```
+ }
> G = rbind(c(0.50, 0.50, 0.00),
+           c(0.00, 0.25, 0.75),
+           c(0.75, 0.00, 0.25))
> stableM(G)
          [,1]      [,2]      [,3]
[1,] 0.4285714 0.2857143 0.2857143
[2,] 0.4285714 0.2857143 0.2857143
[3,] 0.4285714 0.2857143 0.2857143
> 
```

FIGURE 1.3: Proper Values of Matrix A.

1.2 Exercise 2

1.2.1 Écrire une fonction qui permet de trouver les classes d'équivalences d'une chaîne de Markov à partir de sa matrice de transition P .

```
# EXO 2

# Takes a  stochastic matrix of size n x n
equiv <- function(P){
# we create a new matrix T with the same size of P
  m <- nrow(P)
  T = matrix(0 ,m, m)
# Initiating our lopp value i to 1
  i = 1
# We start a while loop from i to m
  while(i<=m){
# We create at each loop a bigger vector of size i
    a <- vector(length=i)
# A zeroes matrix with a single colon and m rows
    b <- matrix(0, ncol = 1, nrow = m)
# At each i iteration we make the value in he matrix index of i = 1
    b[1:i] <- 1
# prev and curr are set to 1 and 0 respectively
    prev <- 1
    curr <- 0
# Nested while loop
    while (prev != curr) {
      prev <- sum(which(b>0))
      n <- nrow(a)
      c <- cbind(sum(P[a,],1))
      d <- which(c>0)
      f <- ncol(d)
      o <- matrix(rep(1,n), ncol = 1, nrow = n)
      b[1,d] <- o[1,n]
      newM <- sum(which(b>0))
      a <- b
    }
    T(i,) = b
# increment the i value.
    i=i+1
  }

  F <- t(T)
# C is a matrix of 0s and 1s.
  C <- T&F
# V is a row vector of 0s and 1s. V(i)=1 if the class C(i) is
   closed, and 0 otherwise.
  V <- (sum(t(C) == t(T)) == m)

  return(C)
}
```

Algorithm for finding C(i)

The following algorithm partitions a finite set of states S into communicating classes. Let m denote the number of elements in S.

1. For each i in S, let $T(i) = i$;

2. For each i in S, do the following: for each k in $T(i)$, add to $T(i)$ all states j such that $Pkj > 0$. Repeat this step until the number of elements in $T(i)$ stops growing. When there are no further elements to add, we have obtained to-sets $T(i)$ for all the states in S. A convenient way to express the set $T(i)$ is as a row vector of length m of 0s and 1s, where the jth entry is 1 if j belongs to $T(i)$ and 0 otherwise. Viewed this way, we have just constructed an m-by-m matrix T of 0s and 1s such that $T(i,j) = 1 if iß j$, and 0 otherwise.

3. To obtain $F(i)$ for all i, first define the m-by-m matrix F equal to the transpose of T. In other words, $F(i,j) = T(j,i)$. Thus, the ith row of F is a vector of 0s and 1s and an entry 1 at position j indicates that state i can be reached from state j.

4. Now defined C as the m-by-m matrix such that

$$C(i,j) = T(i,j)F(i,j).$$

Notice that $C(i,j)$ is 1 if j is both in the to-set and in the from-set of i, and it is 0 otherwise.

5. The class $C(i)$ is now the set of indices j for which $C(i,j) = 1$. The class is closed exactly when $C(i) = T(i)$.[1]

1.3 Exercise 3

1.3.1 Écrire une fonction qui à partir de la matrice de transition P , détermine la distri-bution invariante.

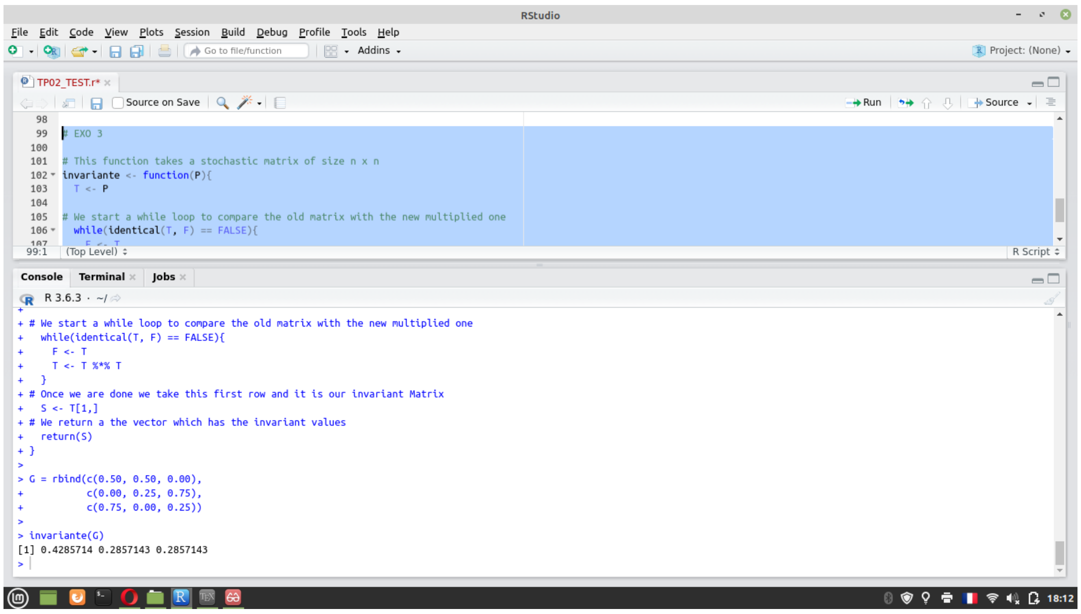
```
# EXO 3

# This function takes a stochastic matrix of size n x n
```

```r
invariante <- function(P){
  T <- P

# We start a while loop to compare the old matrix with the new
    multiplied one
  while(identical(T, F) == FALSE){
    F <- T
    T <- T %*% T
  }
# Once we are done we take this first row and it is our invariant
    Matrix
  S <- T[1,]
# We return a the vector which has the invariant values
  return(S)
}
```

The proper values of the matrix are as follow.



```
# EXO 3

# This function takes a stochastic matrix of size n x n
invariante <- function(P){
  T <- P

# We start a while loop to compare the old matrix with the new multiplied one
  while(identical(T, F) == FALSE){
```

```
+
+ # We start a while loop to compare the old matrix with the new multiplied one
+   while(identical(T, F) == FALSE){
+     F <- T
+     T <- T %*% T
+   }
+ # Once we are done we take this first row and it is our invariant Matrix
+   S <- T[1,]
+ # We return a the vector which has the invariant values
+   return(S)
+ }
>
> G = rbind(c(0.50, 0.50, 0.00),
+           c(0.00, 0.25, 0.75),
+           c(0.75, 0.00, 0.25))
>
> invariante(G)
[1] 0.4285714 0.2857143 0.2857143
>
```

FIGURE 1.4: invariant Matrix